MORRF*: Sampling-Based Multi-Objective Motion Planning

Daqing Yi
Computer Science Dept.
Brigham Young University
Provo, UT 84602, USA
daqing.yi@byu.edu

Michael A. Goodrich
Computer Science Dept.
Brigham Young University
Provo, UT 84602, USA
mike@cs.byu.edu

Kevin D. Seppi
Computer Science Dept.
Brigham Young University
Provo, UT 84602, USA
kseppi@cs.byu.edu

Abstract

Many robotic tasks require solutions that maximize multiple performance objectives. For example, in path-planning, these objectives often include finding short paths that avoid risk and maximize the information obtained by the robot. Although there exist many algorithms for multi-objective optimization, few of these algorithms apply directly to robotic path-planning and fewer still are capable of finding the set of Pareto optimal solutions. We present the MORRF* (Multi-Objective Rapidly exploring Random Forest*) algorithm, which blends concepts from two different types of algorithms from the literature: Optimal rapidly exploring random tree (RRT*) for efficient path finding [Karaman and Frazzoli, 2010] and a decomposition-based approach to multi-objective optimization [Zhang and Li, 2007]. The random forest uses two types of tree structures: a set of *reference trees* and a set of *subproblem trees*. We present a theoretical analysis that demonstrates that the algorithm asymptotically produces the set of Pareto optimal solutions, and use simulations to demonstrate the effectiveness and efficiency of MORRF* in approximating the Pareto set.

1 Introduction

Many tasks assigned to robots are complex, can be performed in several different ways, and must maximize several different performance objectives. For example, a robot in a search task may be expected to maximize the area that it covers while minimizing energy consumption and avoiding risk (see, for example [Mei *et al.*, 2005; Yi and Goodrich, 2014]). As another example, a robot manipulator may need to satisfy performance criteria related to movement, joint velocities, joint accelerations, etc. [Pires *et al.*, 2004].

A common method for finding a solution to a multi-objective optimization problem is to optimize a single objective created by a weighted sum of the multiple objectives. In path-planning the properties of the path produced by this method depend strongly on how each objective is weighted. This means that a programmer, designer, or human teammate must decide how to assign the weights so that the qualitative behavior matches what is intended. In addition to the burden this places on the human operator, optimizing a weighted sum does not work when the multiple objectives are very difficult to compare or are expressed in incommensurate units.

In response to these challenges, it is useful to find *the set of Pareto optimal solutions* to the multi-objective path-planning problem, meaning the set of solutions for which there is no other solution that produces better payoffs for every objective. If an algorithm could produce the set of Pareto optimal solutions then a human could interactively explore this set to find one or more solutions that matches his or her expectations. The objective of this paper is to create an algorithm that efficiently finds the Pareto set in a multi-objective path-planning problem.

Most popular methods in multi-objective optimization do not naturally apply to path-planning problems [Zhang and Li, 2007; Deb and Jain, 2014]. The main reason for this is that path-planning often represents the problem to be solved as a semi-structured tree with an exponential number of possible trajectories through the tree, and the number of evaluations of the objective function required by existing algorithms do not scale well when there are an exponential number of solutions. One approach to addressing this issue is to change the representation for a path by, for example, coding a path as a sequence of fixed-length line segments represented by direction [Ahmed and Deb, 2013; Howlett *et al.*, 2006] or waypoints [Sujit and Beard, 2009; Pires *et al.*, 2004] so that an evolutionary algorithm can be applied. This produces an encoding that can be "fed into" an appropriate evolutionary algorithm to search for the Pareto set. Unfortunately, these approaches do not scale well for large problems, because the number of segments required to represent the paths grows too quickly and estimating the required number of segments *a priori* is very challenging. Another approach is to represent the path as a point in a very high-dimensional vector space. In this approach a path is represented as a point in a $n * d$ dimensional space formed by n d-dimensional way-points. If the number of way-points can be held constant, we can use standard approaches to explore the space. However the search can be more difficult if we allow the number of way-points, and therefor the dimensionality of the optimization problem, to vary. Indeed, we will use this to guide our solution, but the algorithm we present works when the obstacles in the path-planning space introduce discontinuities in these

high-dimensional spaces, which limits the applicability of heuristic-based search approaches [Sujit and Beard, 2009; Zhang and Li, 2007].

The RRT (Rapidly exploring Random Tree) algorithm is popular for finding feasible solutions from a start position to a goal position in continuous or very large search spaces; it also works well when environments have complex obstacles. The reason that RRT is popular is that the tree structure tends to find solutions very efficiently. The RRT* algorithm was a recently introduced modification to RRT that is guaranteed to find an optimal path given enough sampling time [Karaman and Frazzoli, 2011; 2010].

The remainder of the paper presents the MORRF* (Multi-Objective Rapidly exploring Random Forest*) algorithm, which we used to find a set of Pareto optimal paths. MORRF* blends concepts from RRT* a decomposition-based approach to multi-objective optimization [Zhang and Li, 2007].

2 Related Work

Prior work has modeled the search space as a graph and applied a multi-objective A* search to find the solution [Mandow *et al.*, 2005]. The limitation of this approach is that it requires an *a priori* discretization rather than a discretization that is guided by the objectives as is done in RRT*; a coarse discretization throws away potentially valuable information and a fine discretization increases complexity and adds redundancy in the resulting graph structures. Obstacles can make it more difficult to determining which cells in the discretized space are connected to which others, especially when searching a space of more than 2 dimensions such as in planning the trajectory for a robotic manipulator. Another approach that uses an *a priori* discretization (and suffers from these limitations) is to encode a path as a sequence of directions from one cell to next cell and then using the NSGA-II algorithm to find a set of Pareto optimal solutions [Ahmed and Deb, 2013]. Constrained splines have been introduced to interpolate a sequence of way points into a trajectory that avoids obstacles [Ahmed and Deb, 2011], but the effect of the interpolation on the quality of the solution has not been evaluated. In addition to the sorting approach used in NSGA-II, evolutionary algorithms based on the decomposition method have also been proposed [Deb and Jain, 2014].

Evolutionary algorithms can be used to fine the Pareto set, but these approaches tend to be inefficient when applied to spaces with high dimensions [Marler and Arora, 2004]. For such spaces, small deviations in possible solutions may need to be considered in order to find an optimal solution, but this means exploring many possible solutions for problems with non-linearities or multiple local maxima. A path in a fixed-length search tree of depth d can be considered as a point in $\Re^d$, so tree-based approaches followed by an evolutionary "fine-tuning" stage risk running into the problems just listed with evolutionary approaches.

In contrast to searching through and comparing solutions in order to find the Pareto optimal set, decomposition-based methods provide an attractive alternative. In this paper we use a decomposition-based method similar to MOEA-D [Zhang and Li, 2007]. MOEA-D is an algorithm that decomposes a multi-objective optimization problem into a set of subproblems. Each subproblem uses a weighed combination of the objectives to find specific points in the Pareto set or to guide the search for such points. Let $\boldsymbol{\lambda} = [\lambda_1, \cdots, \lambda_K]^T$ be a weighting vector such that $\sum_{k=1}^{K} \lambda_k = 1$. Let $\{c_1(\cdot), c_2(\cdot), \ldots c_K(\cdot)\}$ denote the K-element set of objective functions, let $\boldsymbol{c}(x) = [c_1(x), c_2(x), \ldots, c_K(x)]^T$, and let x denote a potential solution. Finally, let $\boldsymbol{z}^{\text{utop}} = [z_1^*, \cdots, z_K^*]^T$ denote the so-called Utopia reference vector. Three types of decomposition methods have been used in prior work [Zhang and Li, 2007]; however we will use only the two methods described below, leaving the third (the boundary intersection method) to future work.

$$\arg\max_x \sum_{k=1}^{K} \lambda_k c_k(x) \quad \text{weighted sum} \quad (1)$$

$$\arg\min_x \max_{1\le k\le K} \{\lambda_k \left(|c_k(x) - \boldsymbol{z}_k^{\text{utop}}|\right)\} \quad \text{Tchebycheff} \quad (2)$$

The solutions generated by each method are a subset of the Pareto optimal set.

Sampling-based path planning works effectively in continuous space. The RRT (Rapidly exploring Random Tree) has been one of the most popular tools, which efficiently explores the space by randomly sampling the search space; this algorithm tends to work well in the presence of complex obstacles. Unfortunately, RRT has been shown to fail in optimality guarantee [Karaman and Frazzoli, 2010]. In response, the RRT* algorithm was proposed, which uses a *Rewire* process to gradually update the tree structure when new samples of the space indicate that this is needed. Thus RRT* is asymptotically optimal [Karaman and Frazzoli, 2010; 2011].

3 Multi-Objective Rapidly exploring Random Forest*

In this section, we present an algorithm that explores the solution space using RRT*-based tree structures but uses multiple trees in the spirit of decomposition-based multi-objective optimization. Because a set of trees are constructed in the exploration process, we call the algorithm MORRF* (Multi-Objective Rapidly exploring Random Forest*).

Consider a multi-objective path planning problem defined on a bounded, connected open set $X \subset \mathbb{R}^d$ of possible solutions, and K different objectives $\{c_1(\cdot), c_2(\cdot), ...c_K(\cdot)\}$. Without loss of generality, assume that the objective is to minimize these functions. Since the Pareto optimal set is not enumerable, the *goal is to find a representative, finite (M-element) subset of the Pareto optimal set*.

Definition 1. ***Multi-Objective Path Planning*** *Consider a bounded, connected open set $X \subset \mathbb{R}^d$, an obstacle space X_{obs}, an initial state x_{init}, and a goal region X_{goal}. Consider the set of K objectives determined by a vector function $\mathbf{c}(\cdot) = [c_1(\cdot), \ldots, c_K(\cdot)]^T$ defined by $\mathbf{c} : \mathbb{X} \to \mathbb{R}^K$. Denote the obstacle-free space by $X_{free} = X \setminus X_{obs}$. Note that $\mathbf{c}$ is defined for all points in X both those in free space and obstacle space.*

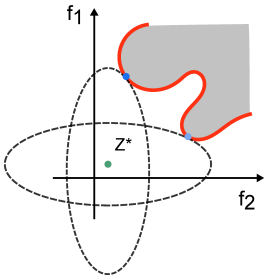


Figure 1: Tchebycheff method of finding Pareto front.

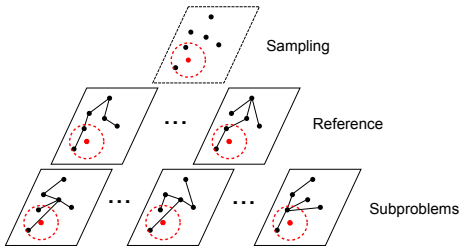


Figure 2: Rapidly exploring process

Define a path *in X as a continuous curve parameterized by s, denoted by $\sigma : [0, s] \to X$. Define the cost of the path as the vector-valued function $\mathbf{c}(\sigma) = \int_\sigma \mathbf{c}(x)ds$. The goal is to find M Pareto optimal paths $\sigma^* \in \Sigma^*$ that (a) $\forall \tau \in [0, s], \sigma^*(\tau) \in X_{free}$; (b) $\sigma^*(0) = x_{init}$ and $\sigma^*(s) = X_{goal}$; (c) There does not exist σ that $\forall k \in K, c_k(\sigma) \leq c_k(\sigma^*)$ and $\exists k' \in K, c_{k'}(\sigma) < c_{k'}(\sigma^*)$.*

Adopting the idea from the MOEA-D algorithm [Zhang and Li, 2007], the M elements in the solution set Σ^* will be obtained by decomposing the multi-objective problem into M subproblems. In this paper, we use the Tchebycheff approach from MOEA-D. The Tchebycheff approach requires us to define a Utopia reference vector $\boldsymbol{z}^{\text{utop}}$ in the fitness space. As illustrated in Figure 1, the Utopia reference vector is defined as that point in cost space that would be obtained if it were possible to find a solution that produced the minimum value for all objectives, that is the k^{th} element of $\boldsymbol{z}^{\text{utop}}$ is given by $\boldsymbol{z}_k^{\text{utop}} = \arg\min_{x \in X} c_k(x)$.

We will need one type of RRT* structure to explore in an attempt to find the Utopia reference vector in payoff space and another type of RRT* structure to find paths that minimize the Tchebycheff condition. Thus, there are two types of tree structures used for the optimization process.

- Each *reference tree* explores using a single objective $c_k(x), k \in K$. The cost of each vertex is calculated using the k^{th} objective function.
- Each *subproblem tree* explores a subproblem $g_m(x \mid \lambda_m, \boldsymbol{z}^{\text{utop}}), m \in M$. The cost associated with each vertex is calculated using $g_m(x)$ defined by the corresponding approach.

The K reference trees and M subproblem trees constitute the exploration forest.

The main flow of the MORRF* algorithm is given in Algorithm 1. Each reference and subproblem tree are a collection of edges and vertices, $G_r = (V_r, E_r)$ and $G_s = (V_s, E_s)$, respectively, and the collection of reference trees and subproblem trees are denoted by $\mathbf{G}_r = \{G_r : r \in \{1, \ldots, K\}\}$ and $\mathbf{G}_s = \{G_s : s \in \{q, \ldots, M\}$.

Note that each tree, reference and subproblem, uses the same set of vertices, meaning they all share the same points in configuration space. The differences between the trees is the edge set; each reference tree and each subproblem tree has a different way of connecting the vertices.

In each iteration, x_{rand} is generated by randomly sampling from the configuration space. The set of vertices is then searched to find that vertex whose position is nearest to the random point; since all trees share the same set of vertices, any tree $G \in \mathbf{G}_r \cup \mathbf{G}_s$ may be used to find the nearest point. The location of this vertex is labeled $x_{nearest}$. The process of finding x_{new} is represented in the top layer of Figure 2.

The exploration at each iteration is given in Algorithm 1. Like RRT*, when the algorithm stops, each reference tree and subproblem tree returns a path, and the set of all these paths forms the solution set.

Algorithm 1 Multi-Objective Rapidly exploring Random Forest*

for each $V_r \in \mathbf{V}_r$ **do**
  $V_r \leftarrow \{x_{init}\}$; $E_r \leftarrow \emptyset$; $i \leftarrow 0$
for each $V_s \in \mathbf{V}_s$ **do**
  $V_s \leftarrow \{x_{init}\}$; $E_s \leftarrow \emptyset$; $i \leftarrow 0$
while $i < N$ **do**
  $x_{rand} \leftarrow$ SAMPLE (i) ; $i \leftarrow i + 1$
  G is arbitrary graph from $\mathbf{G}_r \cup \mathbf{G}_s$.
  $x_{nearest} \leftarrow$ NEAREST(G, x_{rand})
  $x_{new} \leftarrow$ STEER$(x_{nearest}, x_{rand}, \eta)$
  if OBSTACLEFREE$(x_{nearest}, x_{new})$ **then**
    for each $G_r \in \mathbf{G}_r$ **do**
      $G_r \leftarrow$ EXTEND$_{Ref}$ $(G_r, x_{new}, x_{nearest}, r)$
    for each $G_s \in \mathbf{G}_s$ **do**
      $G_s \leftarrow$ EXTEND$_{Sub}$ $(G_s, x_{new}, x_{nearest}, s)$

We now define several functions, using appropriately modified definitions from [Karaman and Frazzoli, 2010].

- SAMPLE(): Returns independent uniformly distributed samples from X_{free}.
- NEAREST(): Returns a position of the vertex whose position is closest to point x. NEAREST$(G = (V, E), x)$ = $\arg\min_{v \in V} \|x - v\|$.
- STEER(): Given two points x and y, returns a point z on the line segment from x to y that that is no greater than η from y. STEER(x, y, η) = $\arg\min_{z \in \mathbb{R}^d, \|z-x\| \leq \eta} \|z - y\|$.
- OBSTACLEFREE(x, x'): Returns True if $[x, x'] \subset X_{free}$, which is the line segment between x and x' lies in X_{free}.

As illustrated in Figure 2, second layer, edges to the reference trees are added before the edges to the subproblem trees. This allows us to compute the Utopia reference vector using the path costs for each reference tree, each reference tree returning a path that approximates the minimum cost for

one objective. The Utopia reference vector is then used to determine which edges should be added for each subproblem.

Consider the second layer in Figure 2, which shows the exploration process for the reference trees. When a new position is obtained (red dot in Figure 2), all reference trees add a vertex that corresponds to this new location. Each reference tree then connects this new vertex to existing nodes by "rewiring" a set of neighboring vertices within a specified radius (red dash circle in Figure 2). The process of rewiring consists of adding edges between existing vertices and the new vertex. This is done using the EXTEND method, given in Algorithm 2.

Algorithm 2 EXTEND$_{Ref}$ $(G, x_{new}, x_{nearest}, k)$

if $x_{new} = x_{nearest}$ **then return** $G = (V, E)$
$V' \leftarrow V \cup \{x_{new}\}$
$x_{min} \leftarrow x_{nearest}$
$X_{near} \leftarrow \text{NEAR}(G, x_{new}, |V|)$
for each $x_{near} \in X_{near}$ **do**
  if OBSTACLEFREE(x_{new}, x_{near}) **then**
    $c'_k \leftarrow \text{COST}_k(x_{near}) + c_k(\text{LINE}(x_{near}, x_{new}))$
    if $c'_k < \text{COST}_k(x_{new})$ **then**
      $x_{min} \leftarrow x_{near}$
$E' \leftarrow E' \cup \{(x_{min}, x_{new})\}$
for each $x_{near} \in X_{near} \setminus \{x_{min}\}$ **do**
  if OBSTACLEFREE(x_{new}, x_{near}) **then**
    $c'_k \leftarrow \text{COST}_k(x_{new}) + c_k(\text{LINE}(x_{new}, x_{near}))$
    if $c'_k < \text{COST}_k(x_{near})$ **then**
      $x_{parent} \leftarrow \text{PARENT}(x_{near})$
      $E' \leftarrow E' \setminus \{(x_{parent}, x_{near})\}$
      $E' \leftarrow E' \cup \{(x_{new}, x_{near})\}$
return $G' = (V', E')$

The precise definitions of the methods used in the Algorithm 2 are given below.

- NEAR(G, x, η): Returns a set of all vertices within the closed ball of radius r_n centered at x, in which $r_n = \min\{(\frac{\gamma}{\xi_d}\frac{\log n}{n})^{1/d}, \eta\}$. The volume of the ball is $\min\{\gamma\frac{\log n}{n}, \xi_d\eta^d\}$.
- LINE(x, x') : $[0, s] \leftarrow X_{free}$ denotes the path defined by line segment from x to x'.
- COST(v): Returns the cost of the unique path (because G is a tree) from x_{init} to the vertex $v \in V$. COST(x_{init}) = 0.

Consider the third layer in Figure 2, which illustrates how the subproblem trees "rewire" to connect to the new vertex. The Utopia reference vector, $\hat{\boldsymbol{z}}_k^{\text{utop}}$ is defined as the k-dimensional vector constructed from each reference tree. The minimum cost of each path from the starting vertex over any other vertex is computed for each reference tree. Using the Utopia reference vector, each subproblem tree connects its new vertex and rewire neighboring vertices in a radius as well. Algorithm 3 precisely follows Algorithm 2 except that instead of computing the cost using one of the objectives, the cost is computed using the Tchebycheff method; each of the m^{th} subproblem trees corresponds to a different weighting vector λ_m. This is performed using the FITNESS method.

Algorithm 3 EXTEND$_{Sub}$ $(G, x_{new}, x_{nearest}, m)$

if $x_{new} = x_{nearest}$ **then return** $G = (V, E)$
$V' \leftarrow V' \cup \{x_{new}\}$
$x_{min} \leftarrow x_{nearest}$
$X_{near} \leftarrow \text{NEAR}(G, x_{new}, |V|)$
for each $x_{near} \in X_{near}$ **do**
  if OBSTACLEFREE(x_{new}, x_{near}) **then**
    $\boldsymbol{c}' \leftarrow \text{COST}(x_{near}) + \boldsymbol{c}(\text{LINE}(x_{near}, x_{new}))$
    $\eta' = \text{FITNESS}(\boldsymbol{c}', \hat{\boldsymbol{z}}^{\text{utop}} \mid \lambda_m)$
    $\boldsymbol{c}_{new} = \text{COST}(x_{new})$
    $\eta_{new} = \text{FITNESS}(\boldsymbol{c}_{new}, \hat{\boldsymbol{z}}^{\text{utop}} \mid \lambda_m)$
    if $\eta' < \eta_{new}$ **then**
      $x_{min} \leftarrow x_{near}$
$E' \leftarrow E' \cup \{(x_{min}, x_{new})\}$
for each $x_{near} \in X_{near} \setminus \{x_{min}\}$ **do**
  if OBSTACLEFREE(x_{new}, x_{near}) **then**
    $\boldsymbol{c}' \leftarrow \text{COST}(x_{new}) + \boldsymbol{c}(\text{LINE}(x_{new}, x_{near}))$
    $\eta' = \text{FITNESS}(\boldsymbol{c}', \hat{\boldsymbol{z}}^{\text{utop}} \mid \lambda_m)$
    $\boldsymbol{c}_{near} = \text{COST}(x_{near})$
    $\eta_{near} = \text{FITNESS}(\boldsymbol{c}_{near}, \hat{\boldsymbol{z}}^{\text{utop}} \mid \lambda_m)$
    if $\eta' < \eta_{near}$ **then**
      $x_{parent} \leftarrow \text{PARENT}(x_{near})$
      $E' \leftarrow E' \setminus \{(x_{parent}, x_{near})\}$
      $E' \leftarrow E' \cup \{(x_{new}, x_{near})\}$
return $G' = (V', E')$

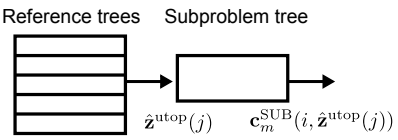


Figure 3: The dependency of the trees in MORRF*.

The FITNESS method computes costs using one of the cost functions in Equations (1)-(2). Different values of λ_m are obtained using the pattern in the MOEA-D algorithm: (a) pre-deterimining the range of the K-cost functions, $\{c_k() : 1 \ldots K\}$ and (b) sampling from the K-dimensional hypercube defined by these ranges. The M samples from this hypercube can be obtained by either creating a uniform (hyper)-grid or by doing uniform sampling across the space.

4 Analysis

The analysis depends on the following restrictions on the cost functions and obstacle placement required by the RRT* algorithm [Karaman and Frazzoli, 2010]. We claim without argument that the cost functions and obstacle placements used in the simulation studies satisfy the restrictions.

Assumption 1. *(Additivity of the objective functions) For a path constructed by composing two other paths (to create a discontinuous path),* $\forall k \in K, \sigma_1, \sigma_2 \in X_{\text{free}}$, $c_k(\sigma_1 \circ \sigma_2) = c_k(\sigma_1) + c_k(\sigma_2)$.

Assumption 2. *(Continuity of the cost functions) For all* $k \in K$*, the cost function* c_k *is Lipschitz continuous, that is, for all paths* $\sigma_1 : [0, s_1] \rightarrow X_{free}$ *and* $\sigma_2 : [0, s_2] \rightarrow X_{free}$,

there exists a constant $\kappa(k) \in \mathbb{R}_+ \cup \{0\}$ *such that* $|c_k(\sigma_1) - c_k(\sigma_2)| \leq \kappa(k) \sup_{\tau\in[0,1]} \|\sigma_1(\tau s_1) - \sigma_2(\tau s_2)\|$.

Assumption 3. *(Obstacle spacing) There exists a constant* $\delta \in \mathbb{R}_+$ *such that* $\forall x \in X_{free}$, $\exists x' \in X_{free}$ *such that*

- *the* δ*-ball centered at* x' *lies inside* X_{free}*;*
- x *lies inside the* δ*-ball centered at* x'.

Lemma 1. *If the Utopia reference vector satisfies* $\forall k, \sigma\ z_k^{\text{utop}} \leq c_k(\sigma)$*, then any solution of Eq.* (2) *is Pareto optimal.*

Proof. The proof is by contradiction. Let the weighting vector λ be arbitrary subject to $\forall k\ \lambda_k \geq 0$, and let $\sigma^* = \sigma^*(\lambda)$ be a solution given that weighting vector. By definition,

$$\sigma^* = \arg\min_{\sigma} \max_{k\in K} \lambda_k |c_k(\sigma) - z_k^{\text{utop}}|. \tag{3}$$

Assume that the path σ^* is not Pareto optimal. Then there exist another path σ^o that dominates σ^* and the Utopia reference vector that satisfies $\forall k \in K$, $z_k^{\text{utop}} \leq c_k(\sigma)$, it follows that $\forall k \in K, z_k^{\text{utop}} \leq c_k(\sigma^o) \leq c_k(\sigma^*)$ and $\exists k' \in K, z_k^{\text{utop}} \leq c_{k'}(\sigma^o) < c_{k'}(\sigma^*)$. These equations imply

$$\begin{aligned} &\forall k \in K, \quad \lambda_k |c_k(\sigma^*) - z_k^{\text{utop}}| \geq \lambda_k |c_k(\sigma^o) - z_k^{\text{utop}}|; \\ &\exists k' \in K, \quad \lambda_{k'} |c_{k'}(\sigma^*) - z_k^{\text{utop}}| > \lambda_{k'} |c_{k'}(\sigma^o) - z_k^{\text{utop}}|; \end{aligned}$$

which yields the following contradiction to Eq (3):

$$\max_{k\in K} \lambda_k |c_k(\sigma^*) - z_k^{\text{utop}}| > \max_{k\in K} \lambda_k |c_k(\sigma^o) - z_k^{\text{utop}}|.$$

□

Lemma 2. *If* σ^* *is Pareto optimal then there exists a weighting vector* $\boldsymbol{\lambda}$*, where* $\forall k\ \lambda_k \geq 0$ *and* $\sum_{k=1}^K \lambda_k = 1$*, such that* σ^* *is a solution of Eq.* (2)*.*

Proof. This is a proof by construction over cases. When σ^* is Pareto optimal, there exist two cases: (a) $\exists k, c_k(\sigma) = z_k^{\text{utop}}$ and (b) $\forall k, c_k(\sigma^*) < z_k^{\text{utop}}$.

Case (a): $\exists k, c_k(\sigma) = z_k^{\text{utop}}$

Define $P(\sigma^*) = \{j \mid c_j(\sigma^*) = z_j^{\text{utop}}\}$ and let $\overline{P} = \{1, \ldots, K\} \setminus P$. Define the weight vector $\boldsymbol{\lambda}$ as $\forall k \in P(\sigma^*), \lambda_k = \frac{1}{|P|}$ and $\forall k \in \overline{P}(\sigma^*), \lambda_k = 0$. For these weights, Eq. (2) returns a set of solution paths, all of which have the same cost for the k-cost functions when $k \in P$ but different possible costs for $k \in \overline{P}$. σ^* is trivially in this set of solution paths.

Case (b): $\forall k, c_k(\sigma^*) > z_k^{\text{utop}}$

For all k, define the weights as $\lambda_k = \frac{\ell_k}{\sum_{j=1}^K \ell_j}$, where $\ell_k = \frac{1}{|c_k(\sigma^*) - z_k^{\text{utop}}|}$. The Tchebycheff cost (Eq. (2)) becomes

$$g^{te}(\sigma^*) = \max_{k\in K} \frac{|c_k(\sigma^*) - z_k^{\text{utop}}|}{|c_k(\sigma^*) - z_k^{\text{utop}}|} \frac{1}{\sum_{j=1}^K \ell_j} = \frac{1}{\sum_{j=1}^K \ell_j}$$

Given any other path σ, we can represent the Tchebycheff cost as follows:

$$\begin{aligned} g^{te}(\sigma) &= \max_{k\in K} \frac{\ell_k}{\sum_{j=1}^K \ell_j} |c_k(\sigma) - z_k^{\text{utop}}| \\ &= \frac{1}{\sum_{j=1}^K \ell_j} \max_{k\in K} \left| 1 + \frac{c_k(\sigma) - c_k(\sigma^*)}{c_k(\sigma^*) - z_k^{\text{utop}}} \right| \end{aligned}$$

Because σ^* is Pareto optimal, $[\exists k' \in K, c_{k'}(\sigma) > c_{k'}(\sigma^*)] \vee [\forall k \in K, c_{k'}(\sigma) = c_{k'}(\sigma^*)]$ for any σ. As $\forall k, c_k(\sigma^*) \geq z_k^{\text{utop}}$, we have $\forall k, c_k(\sigma^*) - z_k^{\text{utop}} \geq 0$. This implies $\exists k' \in K, \frac{c_{k'}(\sigma) - c_{k'}(\sigma^*)}{c_{k'}(\sigma^*) - z_{k'}^{\text{utop}}} \geq 0$, which, in turn, implies that $\max_{k\in K} \left| 1 + \frac{c_k(\sigma) - c_k(\sigma^*)}{c_k(\sigma^*) - z_k^{\text{utop}}} \right| \geq 1$. Therefore, $g^{te}(\sigma) \geq \frac{1}{\sum_{j=1}^K \ell_j} = g^{te}(\sigma^*)$. Thus, σ^* is a solution to Eq. (2).

□

By Lemma 1 and Lemma 2, we have the following:

Theorem 1. *A path is Pareto optimal if and only if it is a solution to Eq.* (2) *for some weight vector.*

Theorem 1 implies that we can use the Tchebycheff method to find the Pareto set for the multi-objective path-planning problems. The next question that needs to be answered is whether the subproblem tree can find the optimal solution of its assigned subproblem.

The way that the RRT* algorithm works is that it incrementally constructs a tree from a root position. The cost of the path from the position of the root node to the positions of every other node converges to the minimal possible cost between the positions as the number of iterations approaches infinity. We restate this as a lemma, and note that it corresponds exactly to that given for Theorem 22 in [Karaman and Frazzoli, 2010].

Lemma 3. *Given Assumptions 1-3, the cost of the minimum cost path from the root to any vertex in RRT* converges to the optimal cost almost surely.*

Lemma 3 and Theorem 1 imply that each reference tree converges to the optimal path from the root to any node in the tree, including a node arbitrarily close to the goal node. This means that the costs returned by those trees for the path from the start to the goal for the cost function c_k converges to the k^{th} element of the Utopia reference vector $\boldsymbol{z}^{\text{utop}}$. We state this as a lemma.

Lemma 4. *Given Assumptions 1-3, the cost of the minimum cost path from the root to any vertex in* k^{th} *reference tree converges to* z_k^* *almost surely.*

We now turn to the proof that the subproblem trees converge to paths in the Pareto set. The proof of this claim requires that we know $\boldsymbol{z}^{\text{utop}}$ to compute the Tchebycheff cost associated with the cost used in the subproblem. If we knew that the reference trees had already converged to $\boldsymbol{z}^{\text{utop}}$, then we could simply instantiate Lemma 3. Unfortunately, the reference trees are converging at the same time that the subproblem trees are converging. We now address this problem.

Let $\hat{\boldsymbol{z}}^{\text{utop}}(v; i)$ denote the approximate Utopia reference vector for position v on iteration i, estimated by the cost from the root to position x from the k-reference trees. Recall that

the m^{th} subtree attempts to generate a solution to Eq. (2) for a given weight vector λ^m. Let

$$\boldsymbol{c}_m^{\text{SUB}}(\boldsymbol{z}) = \arg\min_{x} \max_{k \in K} \lambda_{m,k} |x_k - \boldsymbol{z}_k| \tag{4}$$

denote the cost vector in m^{th} subproblem tree given the reference vector $\boldsymbol{z}$ and let $\hat{\boldsymbol{c}}_m^{\text{SUB}}(i, \boldsymbol{z})$ denote its estimation at iteration i. A subproblem tree obtains $\hat{\boldsymbol{z}}^{\text{utop}}(v)$ for vertex v in the reference trees and generate the corresponding $\boldsymbol{c}_m^{\text{SUB}}(v; i, \hat{\boldsymbol{z}}^{\text{utop}}(v))$. This forms a cascade structure from the reference trees to the subproblem tree. By Lemma 4, we have the convergence of the reference trees.

We introduce Assumption 4 to get Lemma 5.

Assumption 4. *(Lipschitz continuity)* $\boldsymbol{c}_m^{\text{SUB}}(\boldsymbol{z})$ *in Eq.* (4) *and its estimation* $\hat{\boldsymbol{c}}_m^{\text{SUB}}(i, \boldsymbol{z})$ *are Lipschitz continuous, i.e.* $\|\boldsymbol{c}_m^{\text{SUB}}(\boldsymbol{z}_a) - \boldsymbol{c}_m^{\text{SUB}}(\boldsymbol{z}_b)\| \leq K\|\boldsymbol{z}_a - \boldsymbol{z}_b\|$.

Lemma 5. *Given Assumptions 1-4 , the cost of the solution of* m^{th} *subproblem tree converges to the corresponding cost of the* m^{th} *subproblem* $\boldsymbol{c}_m^*$ *almost surely.*

Proof. By Lemma 4, we have $\lim_{j\to\infty}\|\boldsymbol{z}^* - \hat{\boldsymbol{z}}(j)\| = 0$. By Lemma 3, we have $\lim_{i\to\infty} \hat{\boldsymbol{c}}(i, \hat{\boldsymbol{z}}(j)) = \boldsymbol{c}(\hat{\boldsymbol{z}}(j))$. Thus, $\lim_{i\to\infty}\|\boldsymbol{c}(\boldsymbol{z}^*) - \hat{\boldsymbol{c}}(i, \hat{\boldsymbol{z}}(j))\| = \|\lim_{i\to\infty} \boldsymbol{c}(\boldsymbol{z}^*) - \lim_{i\to\infty} \hat{\boldsymbol{c}}(i, \hat{\boldsymbol{z}}(j))\| = \|\boldsymbol{c}(\boldsymbol{z}^*) - \boldsymbol{c}(\hat{\boldsymbol{z}}(j))\|$.

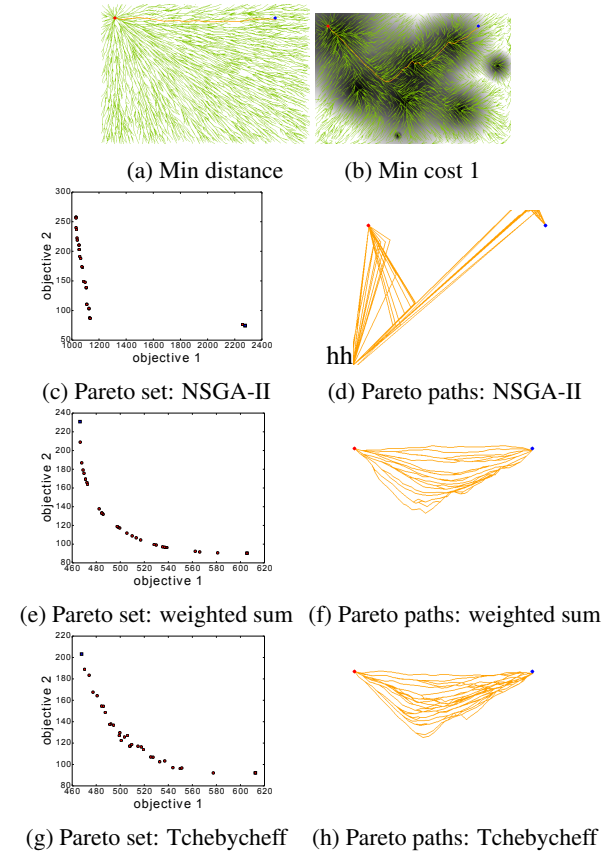
Since $\boldsymbol{c}(\boldsymbol{z})$ and $\hat{\boldsymbol{c}}(i, \boldsymbol{z})$ are Lipschitz continuous; $\lim_{i\to\infty}\|\boldsymbol{c}(\boldsymbol{z}^*) - \hat{\boldsymbol{c}}(i, \hat{\boldsymbol{z}}(j))\| \leq K\|\boldsymbol{z}^* - \hat{\boldsymbol{z}}(j)\|$. As $j \to \infty$, we have $\hat{\boldsymbol{z}}(j) \to \boldsymbol{z}^*$, thus $\lim_{i\to\infty}\|\boldsymbol{c}(\boldsymbol{z}^*) - \hat{\boldsymbol{c}}(i, \hat{\boldsymbol{z}}(j))\| \to 0$. This implies $P(\{\lim_{\substack{i\to\infty \\ j\to\infty}} \boldsymbol{c}_m^{\text{SUB}}(i, \hat{\boldsymbol{z}}(j)) = \boldsymbol{c}_m^*\}) = 1$. □

Now, we can prove that the solution from MORRF* almost surely converges to a subset of the Pareto optimal set.

Theorem 2. *Given Assumptions 1-4 , the solution generated by MORRF* converges to a subset of the Pareto optimal set almost surely.*

5 Simulation

We now present a series of simulation studies that provide evidence that MORRF* produces a representative set of samples from the Pareto set. Results from MORRF* are obtained for path-planning problems with two objectives and three objectives, and are compared to a modified version of the NSGA-II multi-objective path-planning algorithm [Ahmed and Deb, 2013] as well as a variant of MORRF* that uses a weighted sum rather than the Tchebycheff approach. NSGA-II was selected because evidence suggests that it provides more uniform samples from the Pareto set than other approaches [Deb *et al.*, 2002]. We modified the NSGA-II algorithm for this problem to use paths as inputs, represented by a series of waypoints connected by line segments; the cost calculation is identical with that in MORRF*, calling LINE(x_1, x_2) to calculate the cost between two way points x_1 and x_2. The weighted sum approach was chosen because evidence suggests that it works well only when all the objectives are convex [Zhang and Li, 2007] whereas the Tchebycheff approach should bring better diversity in the solutions [Zhang and Li, 2007]. The weighted sum approach uses the same sampling method for weights as that used to generate the λ_i in MORRF*. Each method was run for 5000 iterations and restricted to 30 solutions.



(a) Min distance (b) Min cost 1

(c) Pareto set: NSGA-II (d) Pareto paths: NSGA-II

(e) Pareto set: weighted sum (f) Pareto paths: weighted sum

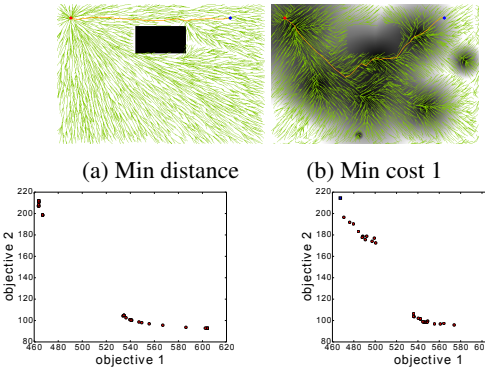
(g) Pareto set: Tchebycheff (h) Pareto paths: Tchebycheff

Figure 4: Path planning with two objectives.

The first simulation study compares three algorithms in an obstacle-free world with two objectives: minimize Euclidean distance, see Figure 4a, and minimize a cost function, see Figure 4b. The first thing to note is that the convergence of NSGA-II-based path-planning is very slow. This is indicated in Figures 4c-4d, which show the approximation to the Pareto set and corresponding paths, respectively, after 5000 iterations; observe how the quality of the paths and sampling of the Pareto set is uneven and unsatisfactory. By contrast, the weighted sum approach returns a set of high-quality solutions close to the Pareto optimal set, see Figures 4e and 4f; Finally, note the somewhat uneven clustering of solutions on Pareto front for MORRF* using weighted sum, and compare this to the slightly more uniform clustering of MORRF* using the Tchebycheff approach in Figures 4g-4h.

We therefore compared results for the two approaches for an environment with obstacles, omitting results for NSGA-II because convergence is so slow. The results are shown in Figure 5. As before, observe that the Tchebycheff approach yields a more uniform sampling, albeit one that appears to be somewhat noisy approximation to the Pareto set.

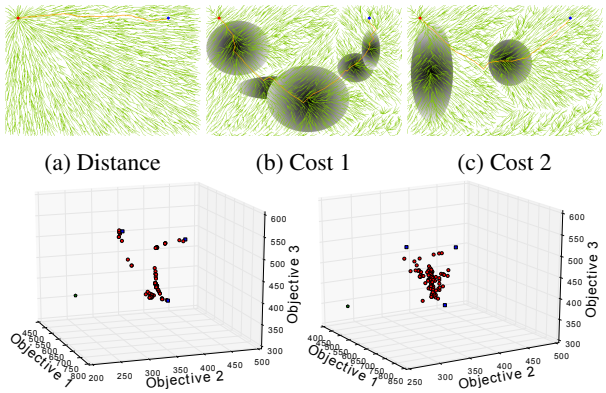
Finally, we evaluated how MORRF* performs three objectives: Euclidean distance and the two other objectives are shown in Figures 6a-6c. As shown in Figure 6, the Pareto front uses the Utopia reference vector (Green point) to better approximate the Pareto set than the weighted sum approach.



(a) Min distance (b) Min cost 1

(c) Pareto set: weighted sum (d) Pareto set: Tchebycheff

Figure 5: Path planning with two objectives and an obstacle.



(a) Distance (b) Cost 1 (c) Cost 2

(d) Pareto set: weighted sum (e) Pareto set: Tchebycheff

Figure 6: Path planning with three objectives.

6 Summary and Future Work

This paper presented the MORRF* algorithm for the multi-objective path-planning problems on continuous spaces. The algorithm blends principles from the RRT* algorithm with principles from multi-objective optimization to produce an algorithm that provides a reasonable approximation of the Pareto set, outperforms a common multi-objective optimization problem on a path-planning problem, and has guaranteed best-case performance.

Future work should extend the algorithm to include not only the weighted sum and Tchebycheff approach but also the boundary intersection approach, which results from [Zhang and Li, 2007] suggest might have even better diversity. MORRF* could also be made more efficient by, for example, using prior information to improve the set of sample points.

Another area of future work is to combine MORRF* with Bellman's principle of optimality. This could be done by setting a goal position as the root node in the algorithm and then generating a set of Pareto optimal paths. The algorithm should then converge to the set of Pareto optimal from any vertex in the tree to the goal.

References

[Ahmed and Deb, 2011] F. Ahmed and K. Deb. Multi-objective path planning using spline representation. In *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pages 1047–1052, Dec 2011.

[Ahmed and Deb, 2013] Faez Ahmed and Kalyanmoy Deb. Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms. *Soft Computing*, 17(7):1283–1299, 2013.

[Deb and Jain, 2014] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, part i: Solving problems with box constraints. *Evolutionary Computation, IEEE Transactions on*, 18(4):577–601, Aug 2014.

[Deb *et al.*, 2002] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[Howlett *et al.*, 2006] Jason K Howlett, Timothy W McLain, and Michael A Goodrich. Learning real-time a* path planner for unmanned air vehicle target sensing. *Journal of Aerospace Computing, Information, and Communication*, 3(3):108–122, 2006.

[Karaman and Frazzoli, 2010] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Robotics: Science and Systems (RSS)*, Zaragoza, Spain, June 2010.

[Karaman and Frazzoli, 2011] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. J. Rob. Res.*, 30(7):846–894, June 2011.

[Mandow *et al.*, 2005] L. Mandow, L. Mandow, J. L. P´erez De la Cruz, and J. L. P´erez De la Cruz. A new approach to multiobjective a* search. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, pages 218–223, San Francisco, CA, USA, 2005. Morgan Kaufmann Publishers Inc.

[Marler and Arora, 2004] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[Mei *et al.*, 2005] Yongguo Mei, Yung-Hsiang Lu, Y Charlie Hu, and CS George Lee. Deployment strategy for mobile robots with energy and timing constraints. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2816–2821. IEEE, 2005.

[Pires *et al.*, 2004] E.J.Solteiro Pires, J.A.Tenreiro Machado, and P.B. de Moura Oliveira. Robot trajectory planning using multi-objective genetic algorithm optimization. In Kalyanmoy Deb, editor, *Genetic and Evolutionary Computation GECCO 2004*, volume 3102 of *Lecture Notes in Computer Science*, pages 615–626. Springer Berlin Heidelberg, 2004.

[Sujit and Beard, 2009] P.B. Sujit and R. Beard. Multiple uav path planning using anytime algorithms. In *American Control Conference, 2009. ACC '09.*, pages 2978–2983, June 2009.

[Yi and Goodrich, 2014] Daqing Yi and Michael A Goodrich. Supporting task-oriented collaboration in human-robot teams using semantic-based path planning. In *Proc. SPIE 9084, Unmanned Systems Technology XVI*, volume 9084, 2014.

[Zhang and Li, 2007] Qingfu Zhang and Hui Li. Moea/d: A multi-objective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11(6):712–731, Dec 2007.